01-03-00

A

Please type a plus sign (+) inside this box → [+]

PTO/SB/05 (2/98)
Approved for use through 09/30/00. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No.	042390.P6602
First Inventor or Application Identifier	Charles R. Yount
Title	A METHOD AND APPARATUS FOR GENERATION OF VALIDATION TESTS
Express Mail Label No.	EM560645832US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

1. [X] Fee Transmittal Form *(e.g. PTO/SB/17)*
(Submit an original, and a duplicate for fee processing)
2. [X] Specification *Total Pages* 20
(preferred arrangement set forth below)
- Descriptive title of the Invention
- Cross References to Related Applications
- Statement Regarding Fed sponsored R & D
- Reference to Microfiche Appendix
- Background of the Invention
- Brief Summary of the Invention
- Brief Description of the Drawings *(if filed)*
- Detailed Description
- Claim(s)
- Abstract of the Disclosure

3. [X] Drawing(s) *(35 U.S.C.113)* *Total Sheets* 6
4. Oath or Declaration *Total Pages*
   a. [ ] Newly executed (original copy)
   b. [ ] Copy from a prior application (37 CFR 1.63(d))
   (for continuation/divisional with Box 16 completed)
   i. [ ] DELETION OF INVENTOR(S)
   Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).

NOTE FOR ITEMS 1 & 13: IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28)

5. [ ] Microfiche Computer Program *(Appendix)*
6. Nucleotide and/or Amino Acid Sequence Submission
(if applicable, all necessary)
   a. [ ] Computer Readable Copy
   b. [ ] Paper Copy (identical to computer copy)
   c. [ ] Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

7. [ ] Assignment Papers (cover sheet & document(s))
8. [ ] 37 CFR 3.73(b) Statement *(when there is an assignee)* [ ] Power of Attorney
9. [ ] English Translation Document *(if applicable)*
10. [ ] Information Disclosure Statement (IDS)/PTO - 1449 [ ] Copies of IDS Citations
11. [ ] Preliminary Amendment
12. [X] Return Receipt Postcard (MPEP 503)
(Should be specifically itemized)
13. [ ] *Small Entity Statement(s) [ ] Statement filed in prior application, Status still proper and desired
14. [ ] Certified Copy of Priority Document(s)
(if foreign priority is claimed)
15. [ ] Other: . . . . . .

16. **If a CONTINUING APPLICATION**, *check appropriate box, and supply the requisite information below and in a preliminary amendment:*

[ ] Continuation [ ] Divisional [ ] Continuation-in-part (CIP) of prior application No: ____/____

Prior application Information: Examiner ____ *Group/Art Unit:* ____

For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

[ ] *Customer Number of Bar Code Label* *(Insert Customer No. or Attach bare code label here)* *or* [X] *Correspondence address below*

Name	BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP				
Address	12400 Wilshire Boulevard, Seventh Floor				
City	Los Angeles	*State*	California	*Zip Code*	90025
Country	U.S.A.	*Telephone*	(310) 207-3800	*Fax*	(310) 820-5988

Name (Print/Type)	Carol F. Barry, Reg. No. 41,600		
Signature		*Date*	12/30/99

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

Attorney's Docket No. 042390.P6602
Express Mail No. EM560645832US

UNITED STATES PATENT APPLICATION

FOR

A METHOD AND APPARATUS FOR GENERATION OF VALIDATION TESTS

Inventors:

Charles R. Yount
Melvyn A. Goveas

Prepared by:

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard, Seventh Floor
Los Angeles, California 90025
(310) 207-3800

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates generally to validation methods and more specifically to computer implemented validation methods that efficiently generate test programs that satisfy a criterion established by a user or by a system designer.

Description of Related Art

Validation test programs are a series of inputs that are used to verify the functionality of a device such as a microprocessor. Validation of a device may be performed in numerous ways. For example, the device being tested may be simulated by a computer program. Alternatively, the device itself may be tested.

Validation methods include test-based methods, coverage-based validation methods, and other known methods. A coverage-based method, for example, executes a test program that generates new coverage data that is then manually evaluated in view of existing coverage data to determine whether the desired coverage has been reached. Coverage data may be used to gauge the "goodness" of test programs that are used to find "bugs" in a design of a device such as a processor. Coverage data is data that indicates what elements of a given set of conditions were activated during a dynamic or static evaluation of a device under test.

Determining whether the coverage data generated from a test program has achieved the coverage that is necessary is a very labor intensive process. Additionally, a significant amount of effort to redirect a test generator to create a new test program is required.

SUMMARY OF THE INVENTION

In one embodiment, a computer system and a computer-implemented method are disclosed for generating validation test programs. The computer system comprises a processor coupled to a storage device. The storage device has stored therein at least one routine. When the processor executes at least one routine, data is generated. The routine causes the processor to generate and analyze a test program. The routine also generates at least one subsequent test program to be generated until at least one termination criterion is met.

BRIEF DESCRIPTION OF THE DRAWINGS

The features, aspects, and advantages of the invention will become more thoroughly apparent from the following detailed description, appended claims, and accompanying drawings in which:

Figure 1 illustrates an exemplary computer system in accordance with an embodiment of the invention;

Figure 2 illustrates a flow chart of an exemplary creation process for an individual test program in a population in an embodiment of the invention;

Figure 3 illustrates an exemplary creation process for each individual test program in a population in an embodiment of the invention;

Figure 4 illustrates a computer system using shared resources in accordance with an embodiment of the invention;

Figure 5 illustrates displays that may be revealed to a use in accordance with an embodiment of the invention; and

Figure 6 illustrates results of an embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention relates to a method and an apparatus for generating validation test programs that are used to simulate a device by using a computer program before the device has been fabricated or the device itself can be tested. The following detailed description and accompanying drawings are provided for the purpose of describing and illustrating presently preferred embodiments of the invention only and are not intended to limit the scope of the invention.

In one embodiment of the invention, an initial population of test programs is either input or generated. The test program(s) are then stored on the storage device. A test program is then selected for execution. The coverage data generated from the execution of a test program is analyzed to determine whether the desired coverage has been attained. The coverage that is required is typically designated by a user or system designer. Based upon the analysis of the new coverage data, test programs are selected for genetic mutation and/or recombination which is used to create a new test program. This new test program is then executed generating new coverage data to be analyzed. This process is repeated until the required coverage is attained.

Disclosed techniques may be used in various forms of validation, including architectural validation, micro-architectural validation, unit-level validation, external bus validation, etc. More specifically, the techniques used in performing functional validation in digital systems including microprocessors and generating functional test suites for computer program or software systems. Generally, the invention is able to attain the same or higher level of coverage in less time with less human effort compared to traditional methods.

In another embodiment of the invention, new test programs are created by a test generator (TG), by using a genetic operation. A genetic operation includes operations

such as a mutation operation, a cross-over operation, or numerous other suitable operations. A mutation involves at least one test program that whenever an operation is used to generate a new test program, data from the population is selected and used in the algorithm. If the designated coverage is not attained (or some other termination criterion or criteria have not been met), the test program is modified by the TG and the process is repeated. Alternatively, an operator using a monitor may input a new test program, thereby injecting "hints" into the operation. A test program is then generated and executed. New coverage data is created and evaluated to determine whether the desired coverage has been attained.

Figure 1 illustrates one embodiment of a computer system 10 which implements the principles of the present invention. Computer system 10 comprises a processor 17, a storage device 18, and a bus 15. Processor 17 is coupled to the storage device 18 by bus 15. In addition, a number of user input/output devices, such as a keyboard 20 and a display 25, are also coupled to the bus 15. Processor 17 represents a central processing unit of any type of architecture (*e.g.*, the Intel architecture, Hewlett Packard architecture, Sun Microsystems architectures, IBM architectures, etc.), or hybrid architecture. In addition, processor 17 could be implemented on one or more chips. Storage device 18 represents one or more mechanisms for storing data such as population data, coverage data, etc. Storage device 18 may include read only memory (ROM), random access memory (RAM), magnetic disk storage mediums, optical storage mediums, flash memory devices, and/or other machine-readable mediums. Bus 15 represents one or more buses (*e.g.*, AGP, PCI, ISA, X-Bus, VESA, etc.) and bridges (also termed as bus controllers). While this embodiment is described in relation to a single processor computer system, the invention could be implemented in a multi-processor computer system. In addition to other devices, one or more of a network 30 may be

present. Network 30 represents one or more network connections for transmitting data over a machine readable media. The invention could also be implemented on multiple computers connected via such a network.

Figure 1 also illustrates that the storage device 18 has stored therein data 135 and software 136. Data 135 represents data stored in one or more of the formats described herein. Software or computer program 136 represents the necessary code for performing any and/or all of the techniques described with reference to **Figures 2-6**. It will be recognized by one of ordinary skill in the art that the storage device 18 preferably contains additional software (not shown), which is not necessary to understanding the invention.

Figure 2 provides one example of an application of evolutionary computation involving a representative algorithm. It will be appreciated that numerous examples exist and that the invention is not limited to any one example. At operation 100, an optimization problem is defined by the user. To illustrate, a user may define the optimization problem as generating iA32 (32-bit Intel Architecture) test programs that, when executed, reach many new states in a new microprocessor design.

At operation 105, different solutions to the optimization problem are defined by the system designer or user. Each solution is called an individual. Each individual test program may be encoded as an abstract syntax tree (AST). Each AST encodes the structure and data of an iA32 test program. For example, some nodes in the AST contain the instructions to be executed during the test program.

At operation 115, a fitness function is defined by the user or system designer. The number of different types of fitness functions is unlimited. The fitness function

defines the "goodness" of a particular individual. For example, the fitness function of each individual test program may be defined as the number of new states it reaches in the new microprocessor design.

At operation 120, an initial set of individuals is created. This initial set of individuals is referred to as the initial population. For example, an iA32 random test generator is used to create a set of random test programs. Alternatively, the population may start with a set of existing test programs.

At operation 125, the individuals are evaluated in the population according to the fitness function. For each test program, the new microprocessor may be simulated and how many new states that are reached are recorded. This number is referred to as the individual's fitness.

At operation 130, if the termination criterion (or criteria) is satisfied, the process is terminated at 132. For example, if all known states in the new microprocessor have been reached, the process is ended. It will be appreciated that a variety of termination criterion may be input by a user and the claimed invention is not limited by examples provided herein.

At operation 135, some of the better individuals in the population are selected. For example, a user may require that genetic operations be randomly chosen for a mutation or crossover operation. The user may further require that, for example, eight random individuals from the population be selected. From the eight individuals, an individual is chosen that has the highest fitness. (This is known in the art as "tournament" selection.) If a crossover operation is chosen, this process is repeated to select another individual.

At operation 140, new individuals are created by applying one of several genetic operations. These new individuals are then added to the population.

Several genetic operations may be chosen to illustrate this operation. If a mutation operation is selected, some of the iA32 instructions from the selected test program are randomly removed and are replaced with new random instructions. If a crossover operation is selected, some of the iA32 instructions from the first selected test program are randomly removed and are replaced with randomly selected instructions from the second selected test program.

For this newly-created test program, the new microprocessor is simulated and the number of new states that are reached (*i.e.*, its fitness) is recorded. The new test program and its fitness are then placed into the population.

At operation 145, some of the poorer individuals in the population are removed from the population. For example, eight random individuals may be selected from the population. At least one individual that has the lowest fitness among the eight individuals is removed from the population. Operations 125 to 145 are then repeated until the termination criterion or criteria are met.

Figure 3 shows one embodiment of the invention for generating new test programs using coverage data as a mechanism to guide the generation of each test program. Examples of test generation includes a microprocessor test generator that may generate a sequence of microprocessor instructions and data that will be executed by a model of the microprocessor or an actual microprocessor. Another example includes a unit-level test generator that may generate a sequence of electrical signals that may be fed sequentially to the device under test. The process described below is

repeated until a test program is executed which generates coverage data that satisfy the coverage designated by a user or system designer. Although an embodiment of the invention uses coverage data to determine the "goodness" of an individual, it will be appreciated other methods may be used. Data may be processed in series or in parallel when practicing the invention.

Figure 3 further illustrates one embodiment of the invention in which a computer program comprises four modules or routines—a test builder (TB) 204, a test generator (TG) 206, a test analyzer (TA) 208, and a feedback engine (FE) 202. These modules may run on the same computer system or one or more may run on separate computer systems that may be connected through a network.

One embodiment of the invention relates to generating high-coverage validation test suites having the characteristics of using search techniques that rely upon improvement of evolutionary computation methods and evaluating newly generated coverage data using existing coverage metrics as a feedback mechanism to guide these methods. One search technique navigates through a space of potential solutions by evaluating a subset of those solutions based upon the selection of new test programs and using the evaluation to choose new test programs until the desired coverage is found or a criterion or criteria established by a user or a system designer are met. The search technique uses algorithms. The algorithms that may be used are known and include genetic algorithms, genetic programming algorithms, evolutionary programming algorithms, simulated annealing algorithms, neural-net training algorithms, and other suitable algorithms.

Coverage monitors (not shown) may be used to gather or collect coverage data either during or after executing a test program to allow the new coverage data to be

evaluated relative to the existing coverage metrics. Coverage monitors include manually-generated specific-event monitors, silicon-based monitors, automatically-generated coverage monitors, code coverage monitors, or other suitable monitors. Additionally, it will be appreciated that the test program-execution medium can vary from software-based simulation to hardware-accelerated simulation to actual hardware.

The process of evaluating the new coverage data generated from a test program involves comparing the coverage data to the desired coverage designated by a user or system designer. If the desired coverage has not been met, the new coverage data is used as evaluation criterion to guide the process to achieve iterative improvement in order to quickly find the designated coverage. Although there are many types of coverage data, the disclosed techniques use coverage data that is measurable.

Figure 3 further shows that the FE is used in determining whether the population has reached its maximum size or desired size at operation 210. For example, if the population has not met its actual maximum size at operation 210, an empty abstract syntax tree (AST) is created at operation 220. A random microprocessor test generator is used to create random instructions and data to complete the AST at operation 230. At operation 240, a translator is used to translate the AST into an executable test program.

The executable test program may be executed, for example, on an RTL simulation model wherein reporting data is generated at operation 260. Reporting data relates to coverage of a set of microarchitectural events, sequences of microarchitectural events, or any combination thereof. This operation may run on TA 208.

The AST and the corresponding coverage data are placed into the population 270 at operation 280. The AST and corresponding coverage data may replace a portion of the existing data in the population, if necessary. The necessity of replacing a portion of

the data in the population is based upon the maximum or desired size of the population and a replacement algorithm known in the art. The operation is ended at operation 285 provided that the desired coverage has been met. If not, a new generation process for an individual test program in the population is performed looping back to operation 210 followed by the subsequent operations as shown in **Figure 3**.

One AST is selected based upon coverage and subjected to a strategy for mutating it 224. Mutation involves changing a test program by replacing a portion of it by a modified or random portion. Thereafter, operations 230, 240, 260, and so on are followed.

It will be appreciated that a variety of methods may be used to determine what genetic operation is used, and the claimed invention is not limited by any example. One method of selecting a genetic operation indicated to the system may involve a designated percentage of operations, such as 90% of the genetic operations selected must be cross-over. Another method is referred to as adaptive tuning. In this method, the system tracks the genetic operation that provides the most gains in coverage. Each type of genetic operation and its average coverage gain generated are recorded in storage device 18. The genetic operation is then automatically selected that provided the greatest coverage gain. Consequently, the desired coverage may be more quickly achieved by this method of selecting the genetic operation.

The cross-over operation 225 involves combining at least one or more characteristics from at least one individual test program and at least one or more characteristics from at least one other test program. These characteristics are used to form a new AST. Thereafter, operations 240, 260, and so on are followed.

Figure 4 shows another embodiment of the invention in which a distributed computation framework is shown using shared computer resources. Shared resources

reduce the time required to practice the invention. This embodiment of the invention comprises a local execution scheduler (LES), a global execution scheduler (GES), component servers, components and graphical user interfaces. Other configurations also may be used.

1. Local Execution Scheduler (LES)

Each application requires a different sequence of operations to be performed with regard to each test program to find the coverage data for that test program. To perform each operation, the LES requests a component from the GES, uses it, and then releases that component so that another application may use that component. This operation can be performed for one or more test programs. For multiple test programs, the operation may be performed in series or in parallel.

2. Global Execution Scheduler (GES)

The GES is a central process to which all other processes may attach. Resources may be shared by a project group by running one GES. Resources such as components may be shared when two or more applications are attached.

The GES is capable of sending a request to component servers to start components. Once the GES sends a request to the component server to start the components, the components perform application-specific functions such as building or analyzing test programs.

As noted above, FE and LES are part of the computer program. To perform each operation, LES requests a component from the GES. LES uses the component and then releases the component to allow another application to use it. This process is generally performed in series but it can be performed in parallel for multiple test programs.

In **Figure 5**, the GES performs the function of controlling data flow from the various component servers. For example, component server 660, component server 662,

and component server 664 are coupled to GES 675. This allows data to flow to and from these components to GES 675. Coupled to these component servers are a plurality of individual components. For example, component 666 and component 668 are coupled to component server 660. Additionally, component 670 is coupled to component server 662. Component server 672 and component 674 are coupled to component server 664. Component servers start the components when the component servers are requested to do so by the GES.

GES 675 is also coupled to graphical user interface 680 and to FE 605, LES 610, FE 640, LES 650, and a remote GUI 655. A remote GUI may be used to view properties, control applications and the GES for machines other than those upon which these processes were started.

FE 605 and LES 610 form one application program. FE 640 and LES 650 form another application program. Both of these applications are also coupled to a GUI and population data. For example, FE 605 and LES 610 are coupled to GUI 600 and to the population data 620. FE 640 and LES 650 are coupled to GUI 630 and population data 660.

Figure 5 shows displays that may result on the GUI by running a test program. For example, from these displays, a user can input directions to FE, if necessary.

Figure 6 shows that an embodiment of the invention achieves the coverage that is desired in less time and in less computation cycles than conventional methods. The results for the number of test programs are shown for the floating point coverage percentage for the invention compared to a conventional method. The point of 40.9% coverage was achieved in 15,000 test programs by the control run (no feedback system) compared to only 2,584 required by the invention's feedback system. Thus, the feedback system shows an 83% reduction in the amount of effort required to find this

level of coverage. Additionally, running the same number of test programs (15,000) with the invention compared to a conventional method results in achieving 24.7% additional coverage by the invention. Accordingly, the invention improves the chances of finding the desired coverage in a shorter period of time in comparison to conventional methods.

In the preceding detailed description, the invention is described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A computer system comprising:

a storage device coupled to a processor and having stored therein at least one routine, which when executed by the processor, causes the processor to generate data, the routine at least causing the processor to at least,

generate at least one test program; and

analyze the test program; and

generate at least one subsequent test program until at least one termination criterion is met.

2. The computer system of claim 1, further comprising:

a population of data stored in a storage device.

3. The computer system of claim 2, wherein a portion of the population is replaced.

4. A machine readable storage medium containing executable program instructions which when executed cause a digital processing system to perform a method comprising:

(a) generating a test program;

(b) evaluating the test program based upon coverage data;

(c) using the evaluation to select a new test program.

5. The computer system of claim 3, further comprising:

(d) determining whether a population has reached a desired size of the population.

6. The computer system of claim 3, wherein the population has not reached the desired size, the method further comprising:

(e) creating an abstract representation of a functional test program.

7. The computer system of claim 6, wherein the abstract representation is translated into a functional test program.

8. The computer system of claim 7, further comprising:

(f) executing at least one test program; and

(g) generating coverage data.

9. The computer system of claim 8, further comprising:

(h) storing abstract representation and corresponding coverage data into a storage device.

10. The computer system of claim 9, wherein if desired coverage has not been achieved, operations (a) through (h) are repeated.

11. A method comprising the computer-implemented operations of:

determining a population size, a first logic which if the population has not reached a designated size, then a representation of a test program is randomly generated, a second logic which if the population has reached a designated size, then a genetic operation is chosen and select at least one test program from a population; and

modify the test program(s) using the genetic operation to create at least one new test program;

executing the new test program(s);

measuring coverage data; and

placing the new test program(s) and coverage data into the population.

12. The method of claim 11, wherein the genetic operation is a mutation; and choosing one test program based upon coverage.

13. The method of claim 12, further comprising:
replacing a portion of the population.

14. The method of claim 13, wherein the genetic operation is a crossover operation, and
choosing two test programs.

15. The method of claim 14 further comprising:
performing a crossover operation.

16. A computer implemented method comprising:
determining a population size, a first logic that when population size has not attained its desired size, then an empty abstract syntax tree is created and a second logic that if the population has attained its desired size then a genetic operation is chosen,
filling the abstract syntax tree with application-specific node types;
translating the abstract syntax tree into an application-specific test program;
executing the test program by the computer processor and generating coverage data; and
placing the abstract syntax tree and corresponding coverage data into a population.

17. The method of claim 16, wherein choosing a genetic operation further comprises:
choosing a mutation operation.

18. The method of claim 17, further comprises:
choosing at least one abstract syntax tree; and

replacing a portion of the coverage data.

19. The method of claim 17, further comprising:

choosing at least two abstract syntax trees; and

performing a crossover operation.

20. A computer implemented method comprising:

determining a population size;

choosing a genetic operation;

choosing two abstract syntax trees based upon coverage data;

performing a genetic operation to form a new abstract syntax tree;

translating the abstract syntax tree into an application-specific test program;

executing the test program;

generating coverage data; and

putting the abstract syntax tree and corresponding coverage data into a population.

ABSTRACT OF THE DISCLOSURE

A computer system and a computer-implemented method for generating test programs that satisfy at least one termination criterion. The computer system includes a hardware unit to transmit data. A processor is coupled to the hardware unit and to a storage device. The storage device has stored therein at least one algorithm and a plurality of routines. When the processor executes a routine(s), data is generated. The routine causes the processor to access an algorithm, generate a test program, and analyze a test program. A computer implemented method is also disclosed for generating test programs.

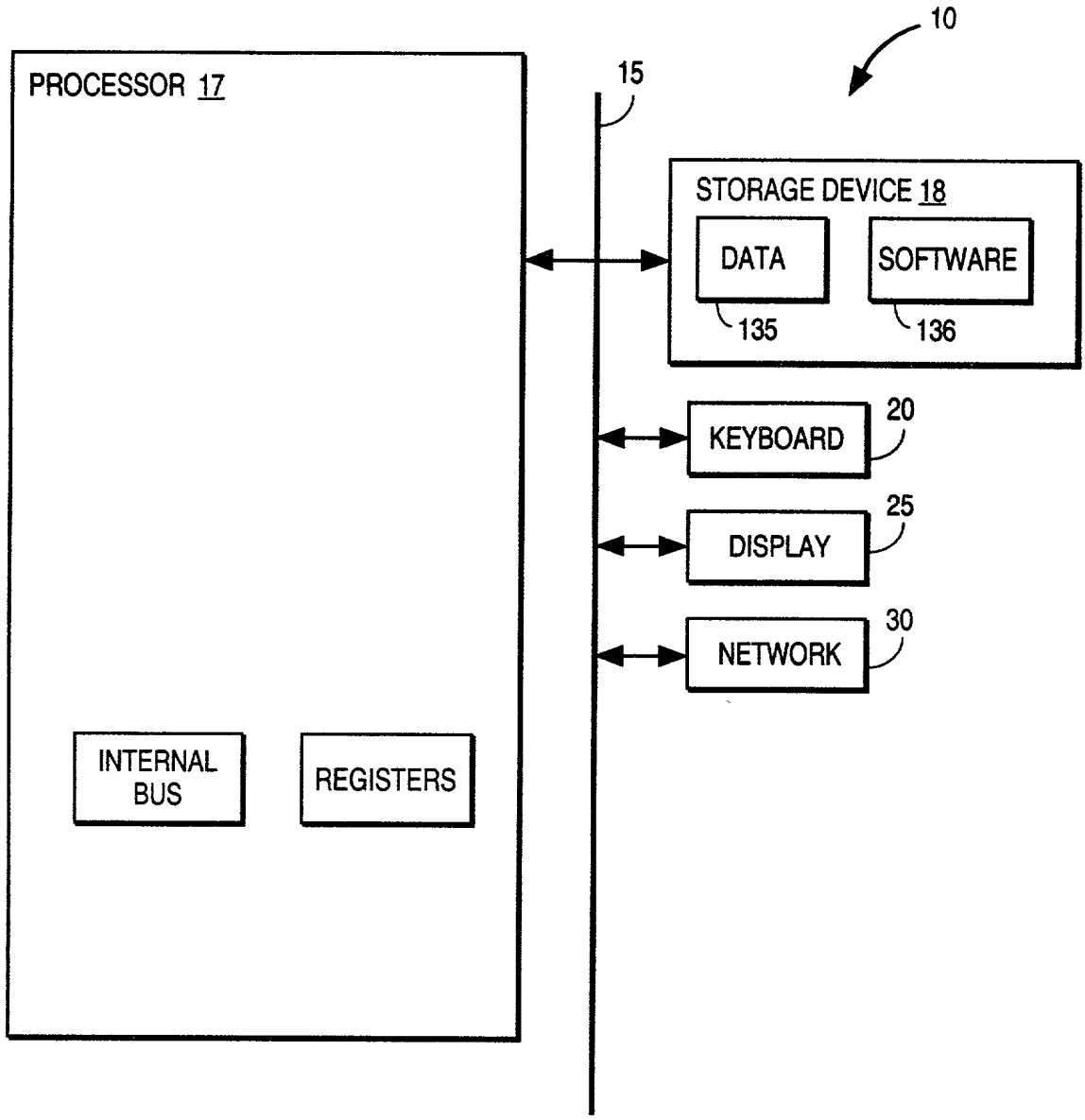
10
PROCESSOR 17
15
STORAGE DEVICE 18
DATA
135
SOFTWARE
136
KEYBOARD
20
DISPLAY
25
NETWORK
30
INTERNAL BUS
REGISTERS

FIG. 1

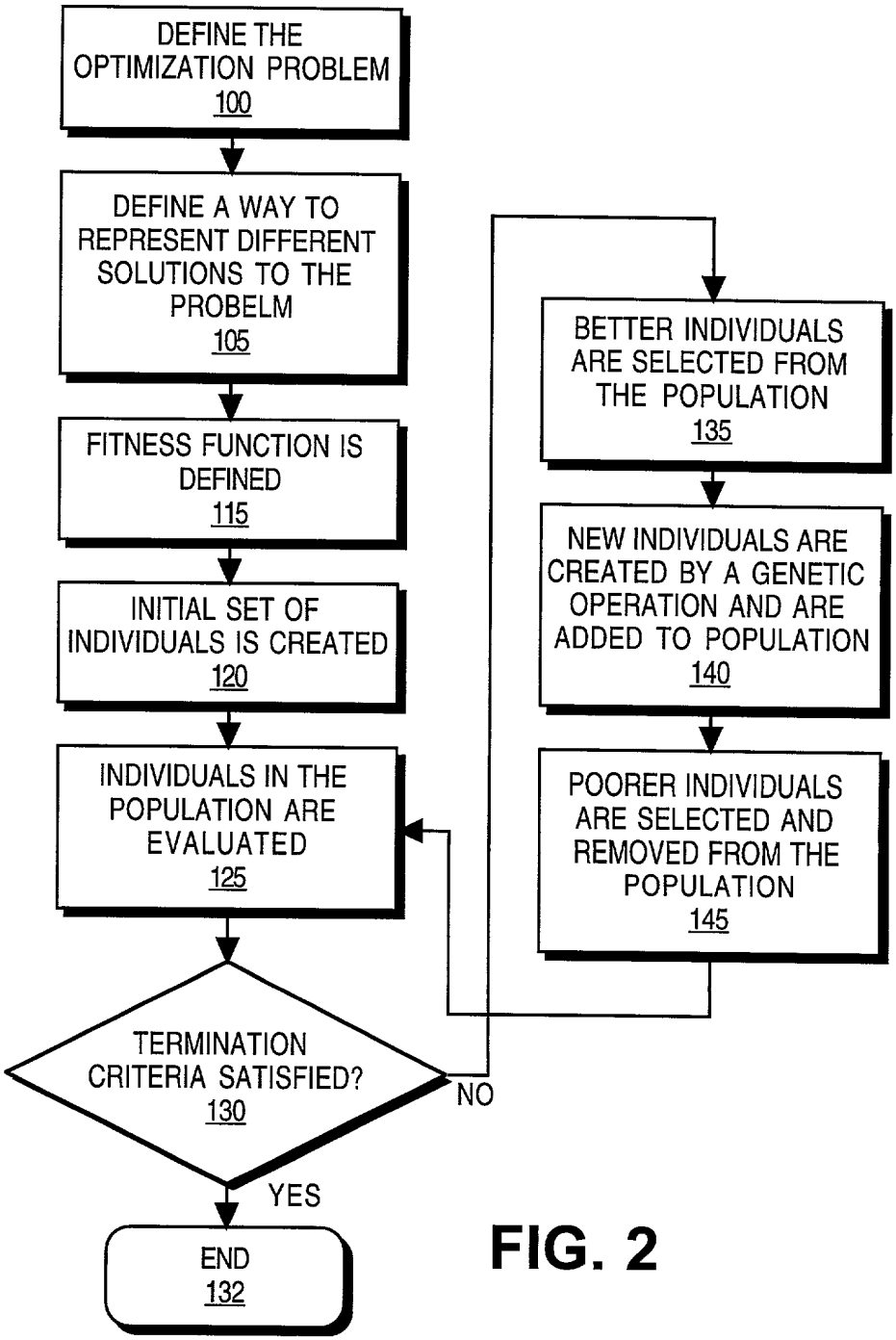

FIG. 2

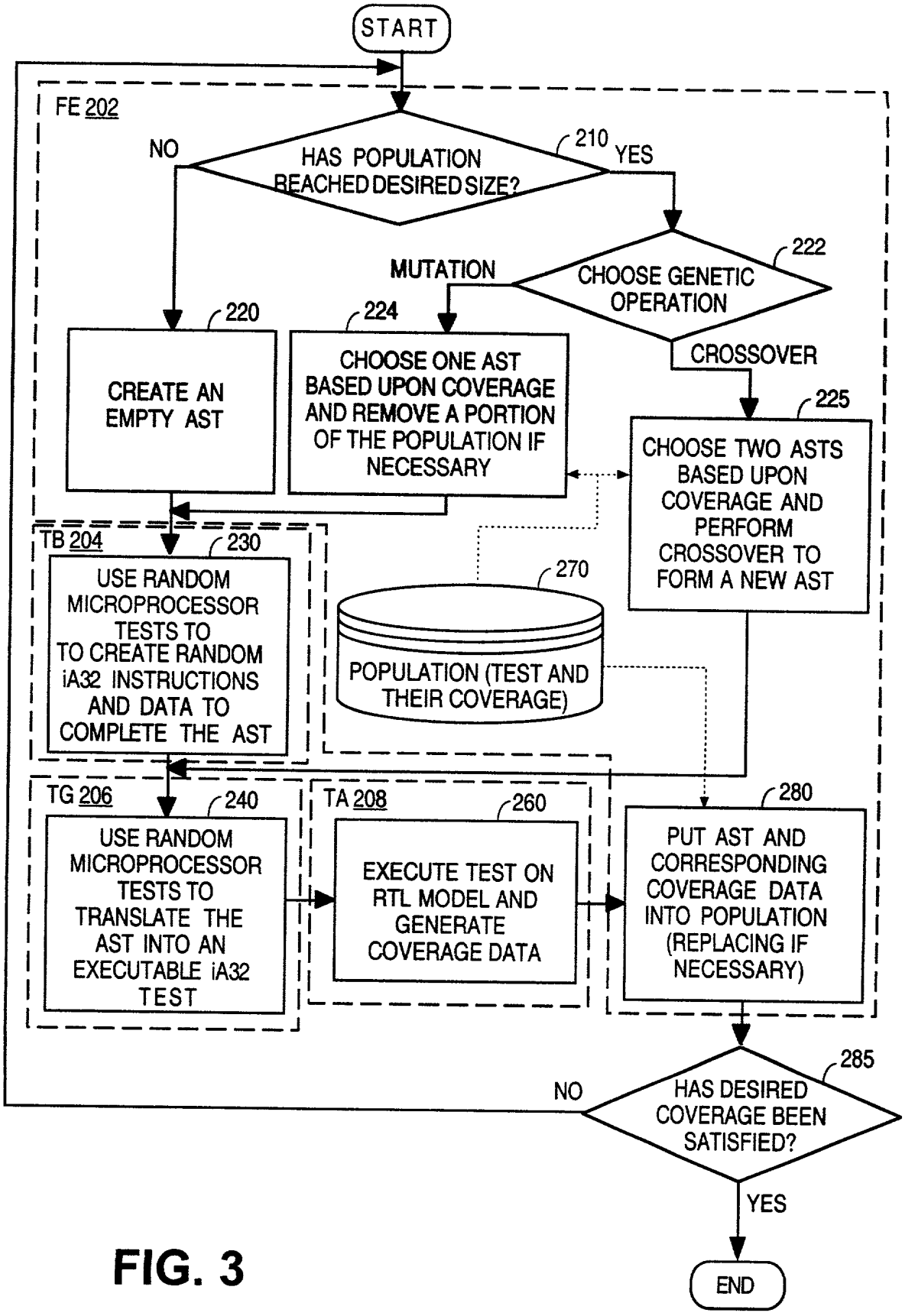

FIG. 3

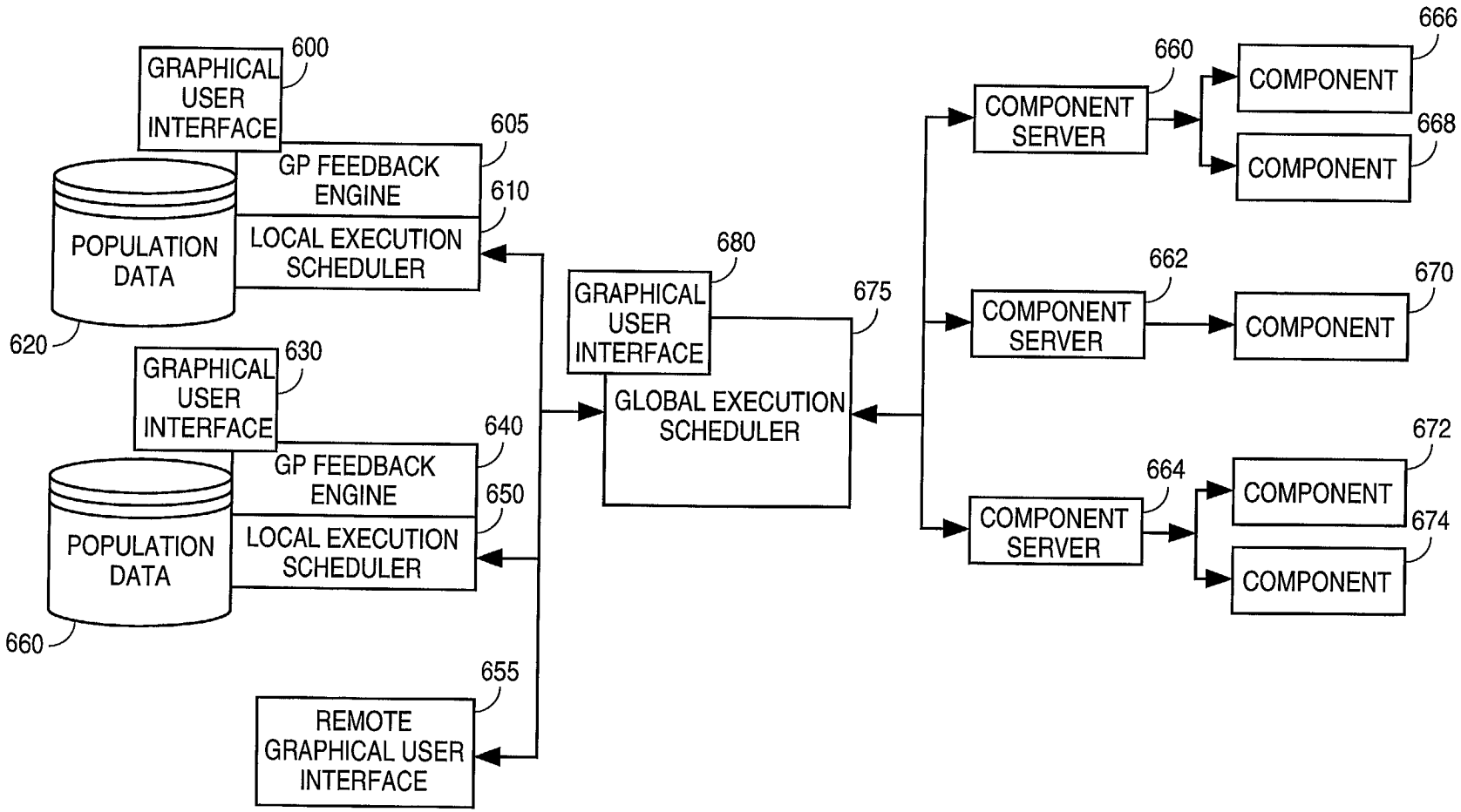
GRAPHICAL USER INTERFACE
600
GP FEEDBACK ENGINE
605
LOCAL EXECUTION SCHEDULER
610
POPULATION DATA
620
GRAPHICAL USER INTERFACE
630
GP FEEDBACK ENGINE
640
LOCAL EXECUTION SCHEDULER
650
POPULATION DATA
660
REMOTE GRAPHICAL USER INTERFACE
655
GRAPHICAL USER INTERFACE
680
GLOBAL EXECUTION SCHEDULER
675
COMPONENT SERVER
660
COMPONENT SERVER
662
COMPONENT SERVER
664
COMPONENT
666
COMPONENT
668
COMPONENT
670
COMPONENT
672
COMPONENT
674

FIG. 4

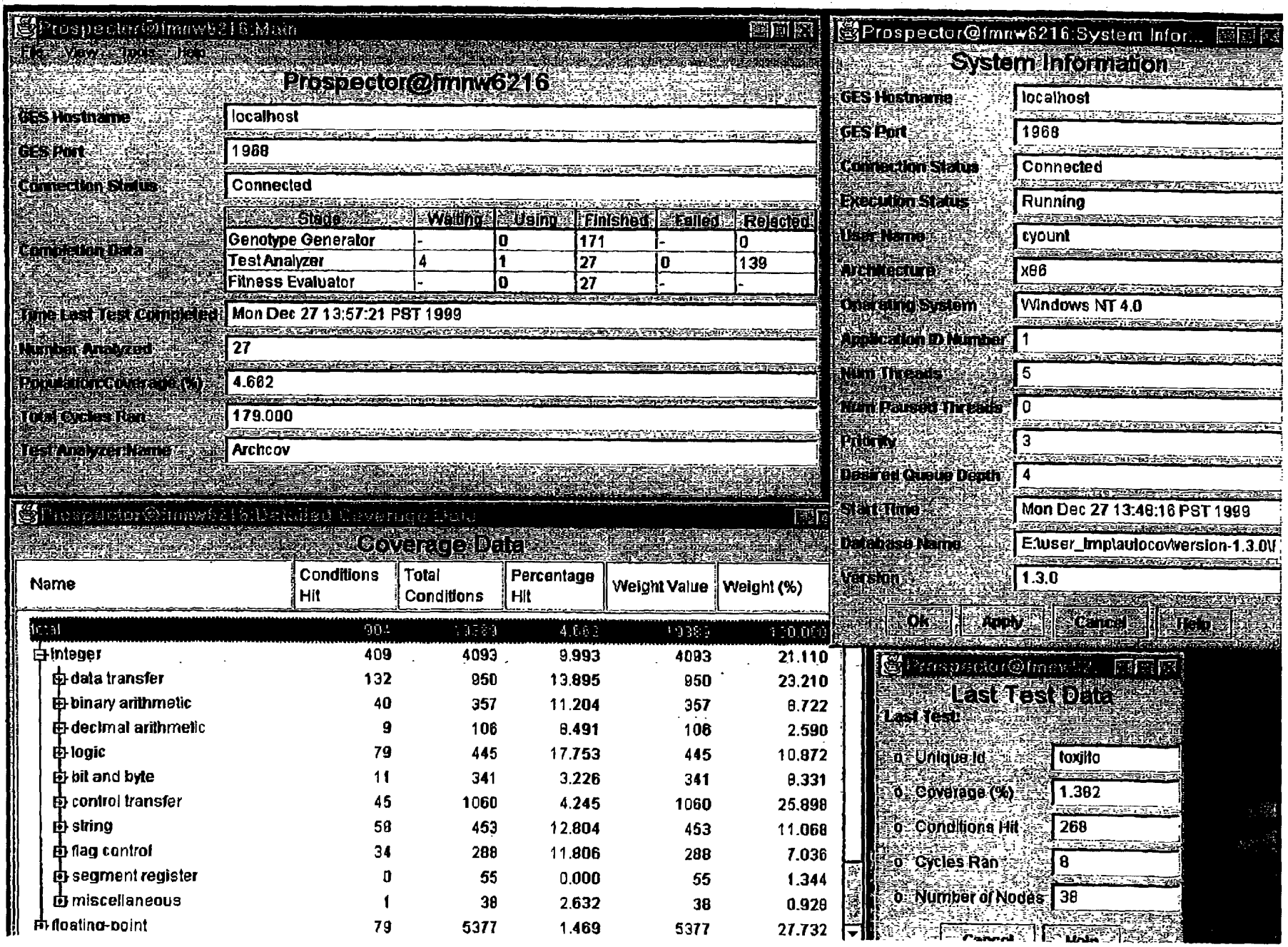

FIG. 5

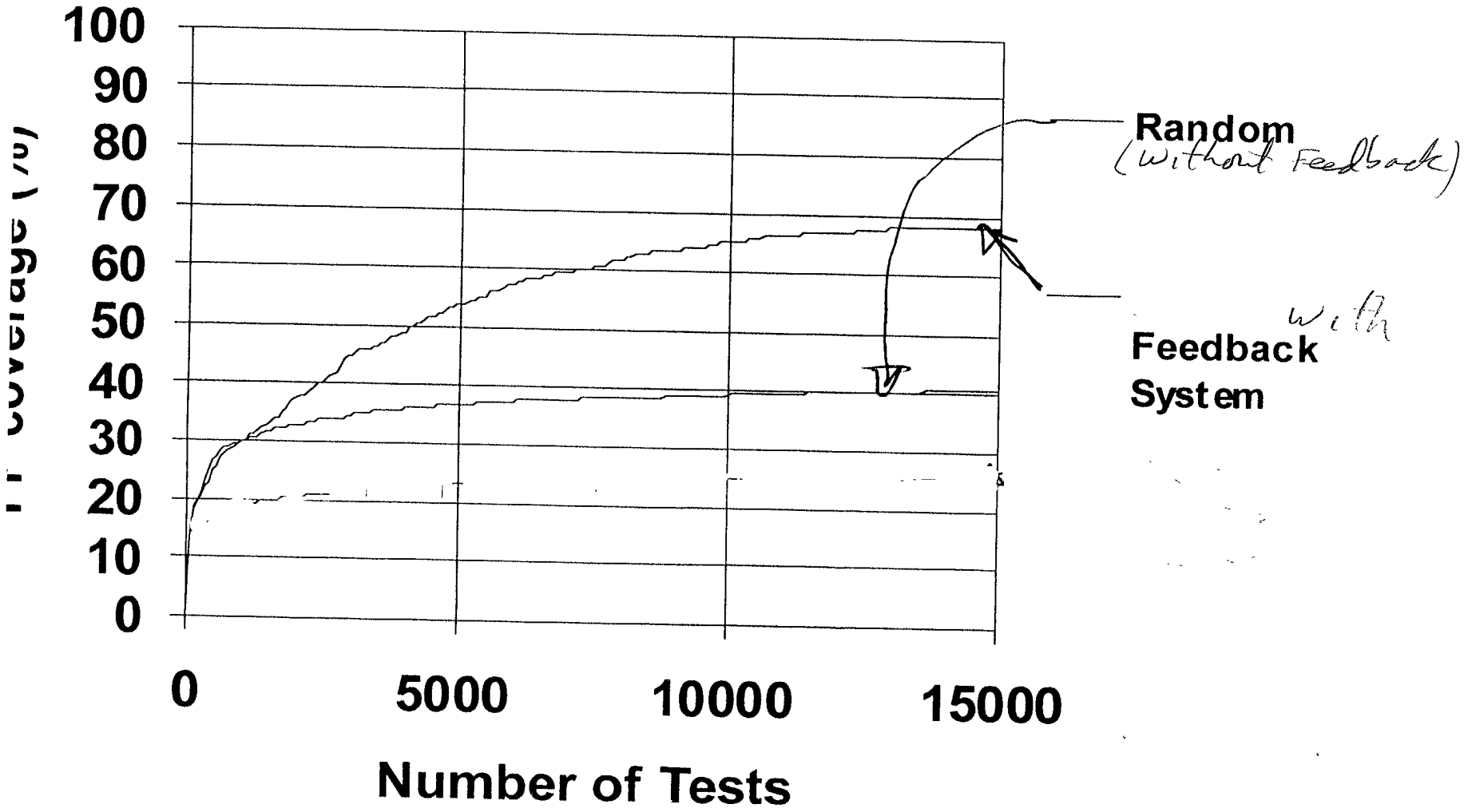

FIG. 6